Running CFU example

- In /home/riscv/wsp/neorv32/rtl/core/neorv32_top.vhd
  - RISCV_ISA_Zxcfu   : boolean   := true;
- In /home/riscv/wsp/neorv32/rtl/core/neorv32_package.vhd
  - RISCV_ISA_Zxcfu   : boolean   := true;
- Generate bitstream
- Program FPGA
- Cd to cfu example
  - Make clean
  - Make exe
- Download the executable with cutecom

Running CFU example

- Desired output:

```
Booting from 0x00000000...

<<< NEORV32 Custom Functions Unit (CFU) - Custom Instructions Example >>>

[NOTE] This program assumes the default CFU hardware in
     'rtl/core/neorv32_cpu_cp_cfu.vhd' that implements
     the Extended Tiny Encryption Algorithm (XTEA).

XTEA key: 0x207230ba1ffba710c45271efdd01768a

XTEA SW encryption (40 rounds, 64 words)...
XTEA HW encryption (40 rounds, 64 words)...
Comparing results... OK

XTEA SW decryption (40 rounds, 64 words)...
XTEA HW decryption (40 rounds, 64 words)...
Comparing results... OK

Execution timing:
ENC SW = 68517 cycles
ENC HW = 12608 cycles
DEC SW = 93539 cycles
DEC HW = 12614 cycles
Average speedup: ~6x

Executing non-implemented CFU instruction (raise ILLEGAL INSTRUCTION exception)...
<NEORV32-RTE> [cpu0|M] Illegal instruction @ PC=0x0000069C, MTINST=0x00F7F78B, MTVAL=0x00000000 </NEORV32-RTE>

CFU demo program completed.
```

Running CRC example

- In /home/riscv/wsp/neorv32/rtl/core/neorv32_top.vhd
  - IO_CRC_EN : boolean := true
- In /home/riscv/wsp/neorv32/rtl/core/neorv32_package.vhd
  - IO_CRC_EN : boolean := true
- Generate bitstream
- Program FPGA
- Cd to CRC example
  - Make clean
  - Make exe
- Download the executable with cutecom

Running CRC example

- Desired output

```
CMD:> e
Booting from 0x00000000...


<<< CRC Unit Demo Program >>>

Test string: '12345678'

[CRC8]
Polynomial = 0x00000007
Seed       = 0x00000000
Result     = 0x0000005b [OK]

[CRC16]
Polynomial = 0x00001021
Seed       = 0x00000000
Result     = 0x000096b9 [OK]

[CRC32]
Polynomial = 0x04c11db7
Seed       = 0xffffffff
Result     = 0xf58d7b78 [OK]

Program completed.
```

Running CRC with vs without the unit

- Use the sw_crc.c file from Eclass to implement a software crc and compare the clock cycles needed for each format (8,16,32 bits) for sw vs hw